Universidad Zaragoza
1542

Trabajo Fin de Grado

Optimización de ruta de reparto con almacenaje

Delivery route optimization with storage

Autor

Kirenia Castellón Carballo

Director

Luis Mariano Esteban Escaño

Escuela Universitaria Politécnica La Almunia 2021

Escuela Universitaria
Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

**ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)**

MEMORIA

Optimización de ruta de reparto con almacenaje

Delivery route optimization with storage

425.21.82

Autor: Kirenia Castellón Carballo

Director: Luis Mariano Esteban Escaño

Fecha: 21 de Septiembre de 2021

INDICE BREVE

INDICE DE CONTENIDO

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

1. RESUMEN

Debido al crecimiento del comercio electrónico, las empresas han comenzado a dar más importancia a la planificación estratégica de las rutas de reparto como forma de reducción de costes logísticos. Su correcta implementación hace que sea un factor competitivo dentro de la cadena de valor.

En este Trabajo de Fin de Grado se planteará el abastecimiento de productos en localidades en la Comunidad autónoma de Aragón, con el objetivo de realizar una optimización conjunta de la ruta de transporte ,incluyéndose los costes de compra de los productos y costes de transporte.

Para lograr este objetivo se creará una aplicación web interactiva que calcule los costes totales asociados a cada uno de los municipios a repartir, escogiendo el almacén de origen de la ruta optima de reparto a seguir. Dicha aplicación estará libremente en el servidor Shinyapps.io.

1.1. PALABRAS CLAVE

Traveling Salesman Problem, Costes, Ciudades reparto, Shinyapps, aplicación

2. ABSTRACT

Due to the growth of electronic commerce, companies have begun to give more importance to strategic planning of delivery routes as a way to reduce logistics costs. Its correct implementation makes it a competitive factor within the value chain.

In this Final Degree Project, the supply of products in localities in the Autonomous Community of Aragon will be considered, with the aim of carrying out a joint optimization of the transport route, including the purchase costs of the products and transport costs.

To achieve this objective, an interactive web application will be created that calculates the total costs associated with each of the municipalities to be distributed, choosing the warehouse of origin of the optimal distribution route to follow. This application will be freely on the Shinyapps.io server.

3. INTRODUCCIÓN

La distribución dentro de ciudades y en su entorno, ha adquirido importancia en los últimos años, muchas son las empresas que realizan sus envíos a distancias cercanas, obteniéndose mayores beneficios en consecuencia de una buena planificación estratégica de las rutas. Debido al crecimiento del comercio electrónico y la subida del precio del gasóleo, hace necesario que las empresas optimicen su eficiencia, por lo que se necesitan herramientas dinámicas que recalculen las rutas con el fin de minimizar costes. Los costes de transporte forman parte de uno de los componentes más controlables dentro de la cadena de valor, siendo un factor competitivo.

La reducción de los costes logísticos es uno de los objetivos que se plantean en este trabajo fin de grado, buscando la optimización de la ruta escogida entre los distintos municipios.

Como primer objetivo del trabajo de fin de grado se postula la creación de una aplicación, con una interfaz de fácil manejo, que permita mostrar, con la selección previa del número de unidades y de la ruta a seguir, el coste de transporte mínimo correspondiente a la distancia de la ruta optimizada.

El escenario en que se plantea la optimización es la región de Aragón. Para el problema planteado, se creará una red de abastecimiento que conecte las cabeceras de comarca de la Comunidad Autónoma de Aragón. Asimismo, se creará un catálogo de productos a ser distribuidos por los diferentes núcleos urbanos. Estos productos tendrán un coste distinto según el almacén de origen del reparto donde se ubiquen. Por tanto, se plantea una optimización de la ruta de reparto que dependerá de dos factores, el precio de compra del producto y el coste de transporte.

La metodología propuesta que se emplea para lograr los objetivos propuestos se muestra posteriormente:

1.-Investigacion sobre la ruta optima de reparto utilizada y sobre el problema del viajante (Traveling salesman problem).

2.-Implementacion y ejecución del algoritmo que se base en la reducción del coate de compra de los productos en el almacén de origen.

3.-Creacion de la aplicación web que permita mostrar los destinos de la ruta, escoger el número de unidades de 10 productos y el precio del combustible en ese momento, obteniéndose la optimización referente a la disminución del coste de compra de los productos.

4.- Se obtendrán conclusiones que se reflejarán en la memoria de este trabajo fin de grado.

5.- Propuestas de mejoras/cambios, si son necesarios.

4. DESARROLLO

Se presenta un caso de distribución en la comunidad de Aragón, en la que se centra la distribución de mercancía por carretera a las 33 cabeceras de comarca, planteándose un cálculo de las rutas tratando de minimizar la distancia recorrida y consecuentemente los costes derivados del transporte y la compra del producto.

Nuestro problema reúne un conjunto de ciudades, para hallar un camino cerrado de costo mínimo, visitando cada ciudad exactamente una vez. El costo estará representado por la distancia entre las ciudades, el coste de los productos, en nuestro caso el coste de combustible y el coste fijo de transporte entre ellas. Se tendrá un catálogo de productos en el almacén de origen en el cual se podrá seleccionar la cantidad a enviar.

El problema se aproximará al problema del viajante (Traveling Salesman Problem, TSP), el cual trata de obtener una ruta optima, partiendo de una ciudad de origen, recorrer varias ciudades destino y volver al mismo origen pasando por *n* ciudades. Siendo un problema clásico de optimización combinatoria, cuyo conjunto de soluciones es finito, pero demasiado numerosos para ser tratado de forma directa. Este problema puede aplicarse en muchas situaciones prácticas, tales como el ruteo de vehículos, secuenciales de tareas, conexión de módulos electrónicos, entre otros. Para su resolución de proponen el uso de métodos heurísticos para el cálculos de rutas optimas o cercanas a las óptimas, pues pertenece a la clase de los problemas combinatorios NP-Complejos.

El TSP tiene sus orígenes en el año 1930, dándose a conocer como uno de los problemas de optimización combinatoria más estudiados. Destacando su simple enunciados y su compleja solución en la búsqueda de una respuesta válida para el enunciado propuesto.

Los costes de los productos se cargarán en RStudio en un fichero de Excel. Para el trabajo se generan ejemplos aleatoriamente en RStudio a través del comando "rpois" en el que se crearon valores de costes para cada una de las cabeceras de comarca y para cada uno de los productos.

4.1. DESCRIPCIÓN DEL PROBLEMA

Uno de los problemas que muchas empresas tienen que enfrentar, es la rentabilidad en las rutas de distribución. Por lo tanto, se plantea una optimización conjunta en la ruta de reparto, visualizándose todas las rutas posibles y las ciudades

como próximo destino, tendiendo una visión completa del problema para minimizar las distancia a recorrer en la ruta de transporte y los costes referentes a la compra de los productos.

De esta forma se hará la optimización discreta de la ruta en la que se elegirá cual será el mejor almacén de origen, obteniéndose la mejor ruta de reparto óptima. Pudiendo cambiar el tipo de producto a transportar dependiendo de la modificación que se realice en el archivo Excel previamente.

Se tendrá en cuenta el consumo por kilómetro del camión Mercedes-Benz Atego-815 el cual tendrá un consumo de 19,83L/100km, lo que representará un coste de 22,28 EUR/100Km.

Las tarifas del coste del vehículos ,es un coste fijo por lo que no está incluido dentro de la optimización. en la siguiente tabla:

Km ilimitados por Día

Mínimo 2 días	155€
3 a 7 días	145€

Tabla 1- Precios por kilómetro

Se incluirán entrada numérica para agregar el precio del combustible el día que se realice la entrega de los productos, siendo así más preciso la optimización del coste.

En resumen, se contemplará la posibilidad de que el almacén de origen para el reparto sea cualquier ciudad de la ruta, se estimará para cada origen los costes de compra y los de transporte y se elegirá como almacén inicial de la ruta aquel cuyo costes de compra + costes de transporte sea mínimo.

4.1.1. Historia

Los problemas relacionados con el problema del viajante fueron tratados en el siglo XIX, por el matemático irlandés Sir William Rowan Hamilton y por el matemático británico Thomas Kirkman.

Aparece por primera vez en la década de 1930 en Viena y en Harvard, por Kart Menger, para el este problema podía ser resuelto en un número finito de pasos, pero

en este momento se desconocían la existencia de reglas que pudieran reducir este número. Retomándose años más tarde por Hassler Whitney y Merrill M. Flood en la Universidad de Princeton alrededor del curso 1931-32.

Durante los años 1950-1960, se fue incrementado el uso de este problema entre los científicos de Europa y Estados Unidos. Descubriéndose su formulación como un problema de Programación Lineal en Enteros, solucionándose con planos de corte. Dicho método lo pudo resolver de manera óptima con 49 ciudades mediante la construcción de un recorrido y comprobando que no había uno que pudiera ser más corto.

En los años posteriores continuaron con su estudio investigadores, matemáticos, científicos de la computación, químicos, físicos, entre otros profesionales.

4.1.1.1. Aplicaciones

Por su nombre "el problema del viajero" se le puede atribuir una aplicación muy específica, como decidir la ruta que debe seguir un vendedor. Sin embargo, se puede emplear en cualquier problema que requiera una selección de nodos en cierto orden para reducir costes o distancias.

Muchos de los avances en la historia del TSP se han visto motivadas por las aplicaciones de este, teniéndose en cuenta hoy en día a la hora de transportar mercancías, pasajeros y vehículos en *n* ciudades distintas. Dando lugar a enrutamientos de vehículos, de cableado de ordenadores, recolección de basuras, de autobuses escolares; siguiendo en estos el mismo patrón, pero con diferentes sujetos que realizan la actividad.

También cabe destacarse el uso del TSP en la industria, en las secuencias de los trabajos realizados por las máquinas. Pudiendo realizarse todos los trabajos en cualquier orden, pero con el objetivo de completarlos en el menor tiempo posible.

4.1.2. Formulación

La matriz de costes del problema es la que permite definir el grafo asociado a dicho problema. Los nodos y los caminos se representarán mediante un grafo convexo, con arcos moderados de manera que cada vértice representa una ciudad,

cada arco representa un camino y el peso asociado a cada arco representa la longitud del camino. La solución a este problema se consistirá en encontrar el ciclo Hamiltoniano de menor coste.

Para entender esta formulación, un ciclo hamiltoniano es un camino que visita todos los vértices del grafo una y solo una vez; y el último de los vértices será adyacente al primero; siendo problemas de tipo NP-Complejo. Un ciclo con estas características se considera óptimo.

4.1.2.1. Complejidad NP

Un problema esta dentro del grupo NP, solo puede resolverse mediante un algoritmo no determinístico en tiempo polinomial. Su nombre proviene de "nondeterministic polynomial-bounded", que arbitrariamente escoge una de las posibles acciones entre varias posibilidades, esta elección puede entenderse como una pregunta que se realiza el algoritmo para conducirlo a la solución acertada.

Dentro de los problemas NP, la clase de complejidad NP-complejo abarca los problemas más difíciles de NP.

4.1.2.2. Enfoques del problema NP

Los algoritmos NP-Complejos utilizan un mayor tiempo, respecto a su tamaño de entrada, para resolver el problema. Para llegar a su resolución se utilizan varios enfoques:

- Aproximación: Algoritmo que encuentra sus solución rápidamente pero dentro de cierto rango de error, por lo que esta no necesariamente puede ser correcta.
- Probabilístico: Algoritmo que, en promedio puede obtener una solución al problema que se plantea, para una distribución de los datos entrados.
- Casos Particulares: Estos son casos para los cuales existen soluciones rápidas.
- Heurísticas: Algoritmo que trabaja bien en muchos casos. Generalmente son rápidos, pero no existe forma de medir la calidad de su respuesta.

- Algoritmo genérico: Este algoritmo mejora las posibles soluciones hasta encontrar una que posiblemente se aproxime a la óptima. En este algoritmo tampoco se puede medir la calidad de sus respuesta.

4.1.2.3. Ejemplos NP

Algunos de los problemas en NP que encontramos y a los que haremos frente en este trabajo son:

- Problema del ciclo hamiltoniano; que no es más que un problema que trata de determinar si un ciclo o camino hamiltoniano existe dentro de determinando grafo.
- Problema del viajante de comercio; este problema enuncia la búsqueda del recorrido más corto en el que un viajante de comercio pueda recorrer todas y cada una, volviendo a la ciudad de origen sin pasar dos veces por ninguna de ellas, teniéndose *n* ciudades distribuidas aleatoriamente y separadas por distancias aleatorias. Se podría entender como buscar caminos hamiltonianos.

4.1.3. Algoritmos para su resolución

A continuación, se describirán la serie de algoritmos que en intentan resolver en TSP de varias formas. Cada uno tendrá características diferentes en cuanto a su forma de optimización, tiempo de ejecución y estarán ligados a la resolución del trabajo de fin de grado.

4.1.3.1. Algoritmos voraces

Este método es usado principalmente para resolver problemas de optimización, aproximándose en ocasiones a una solución de problemas que pueden ser computacionalmente difícil, siendo se fácil diseño e implementación, lo que los hace de gran eficiencia.

El método voraz para que arroje una solución dispone de un grupo de datos; y a medida que avanza el algoritmo se van formando dos conjuntos; uno de valores considerados y otro de valores que son rechazados definitivamente. Desarrollando así

una función que seleccione la opción más adecuada con la optimización del camino mínimo.

En un inicio en el conjunto vacío, se va considerando adicionar el mejor valor sin considerar los otros restantes. Al seleccionarse este elemento nuevo, se verifica y se comprueba si puede conducir a la solución. Una vez comprobado que es factible se incorpora al conjunto solución, permaneciendo hasta el final, comprobándose junto al conjunto final y si forma parte de la solución. Finalmente, si lo es termina el algoritmo; sino se realizará este proceso varias veces hasta que se haya considerado cada valor. Es un método rápido, que trata de mirar dos saltos en vez de uno.

Con esto queremos decir que lo fundamental es que cada opción se trata una sola vez, no pudiendo replantearse el problema. Haciendo que el algoritmo sea sencillo y eficiente, pero menos potente.

4.1.3.2. Algoritmos Heurísticos

Se les conoce como algoritmos heurísticos cuando la solución no se determina de forma directa, sino mediante pruebas y ensayos. Estos algoritmos proporcionan una buena aproximación para la solución del TSP.

Todos estos algoritmos siguen un método similar, en un inicio se genera candidatos de posibles soluciones en función de cierto patrón. Posteriormente se someten a pruebas de acuerdo con una pauta que caracteriza a la solución. Se van comprobando los candidatos, en caso de ser rechazado se genera otro, y el anterior no se considera en la solución. Denominándose, "Algoritmos con vuelta atrás".

Los procedimientos presente en este algoritmo son los siguientes:

- Comparación con la solución óptima: En el que se mide la desviación porcentual de la solución heurística frente a la óptima, calculando el promedio de esta desviaciones.
- Comparación con una cota: Aunque se cuente con un número reducido de ejemplos no se puede obtener un algoritmo óptimo. Se puede evaluar en estos casos la calidad del método heurístico con una cota del problema.
- Comparación con otros heurísticos: Se usan en este método algoritmos conocidos, comparándolos con el nuevo, para ello se establecen parámetros de comparación como su tiempo, facilidad de implementación, simplicidad y su flexibilidad. Este método es el más empleado.

4.1.3.3. Métodos iterativos

Este método se basa en tomar, de forma aleatoria, un subtour inicial en el cual a cada paso se le añade un elemento hasta completar toda la solución. En el caso del TSP se le añadirá una nueva ciudad hasta que se complete el recorrido optimo, seleccionándose una ciudad para cada paso que se dé, decidiéndose donde se insertará la ciudad.

4.1.3.3.1. Vecino más próximo

El vecino más próximo (nearest insertion) consiste en construir un ciclo hamiltoniano de bajo coste en el vértice más cercano a un lado, para esto se realizan los siguientes pasos:

- Selección del vértice inicial arbitrariamente, *x1*.
- Búsqueda del vértice más cercano a *x1* yo lo introducimos en el recorrido. Marcándose posteriormente como vértice visitado.
- El subtour se forma por los vértices *x1, x2, x3, …, Xk*, se buscará el Xk+1, siendo el más próximo a *Xk*, formando a ser parte del tour.
- Si se visitan todos los vértices, se da por finalizado el algoritmo, sino volvemos al paso anterior.

4.1.3.3.2. Vecino más lejano(farthest insertion)

Este método se basa en construir un circuito el cual utiliza un determinando de conjunto de vértices ,en el que se van insertando uno a uno los vértices restantes hasta formar un ciclo hamiltoniano. Consistiendo de esta manera en elegir la ciudad o vértice más alejada de las ciudades del circuito actual. Tendrá los pasos vistos anteriormente pero siempre buscando el vértice más alejado.

4.1.3.3.3. Árbol de expansión mínima

En la teoría de grafos, el árbol es un grafo en el que un único nodo es el que permite acceder a los restantes, teniendo cada nodo un único predecesor, excepto el inicial que no tiene. Este árbol se puede definirse como un grafo sin ciclos convexo o

un grafo sin ciclos con *n-1* aristas, siendo *n* el número de vértices, su grado se determinará según el número de subárboles de cada nodo. En este se encontrarán las hojas que serán los nodos finales del árbol.

Cuando el árbol es de máximo alcance, es porque obtenemos un grafo convexo sin ciclos. Del mismo modo, cuando hacemos referencia a un árbol de mínima expansión, es porque la suma de sus aristas es mínima y el alcance del árbol de máximo alcance es mínimo.

4.2. HERRAMIENTAS

4.2.1. R-Project Statistical

Para la realización de este trabajo de fin de grado se ha elegido el programa R-Studio Statistical Computing versión 1.4.1717.

Este programa es un software libre que permite la manipulación de datos, realización de cálculos, así como la obtención de gráficos para su posterior análisis. Como principales características podemos resaltar que presenta un depurador interactivo para diagnosticar y corregir los errores rápidamente. Dentro del programa existe un soporte integrado que nos permite la búsqueda de comandos necesitados.

Este software brinda todo tipo de herramientas estadísticas que permiten realizar diferentes tareas, estando disponibles más de 15303 paquetes que permiten su aplicación en el análisis de datos, grafico de mapas, aplicaciones financieras, etc. Es un software que se encuentra en constante actualización, abierto a que los usuarios contribuyan con sus paquetes básicos.

A continuación, se ilustran los paquetes que han sido utilizados para la realización del trabajo.

4.2.1.1. Readxl versión 1.3.1

Es un paquete diseñado para importar hojas de Excel a R, haciendo que este paquete sea ligero y eficiente, pero sin contar con funciones avanzadas. El mismo funciona en Windows, Linux y OSX, no necesitando a diferencia de otros paquetes la instalación en el sistema operativo, tales como Java o Perl.

Es compatible con hojas de cálculo, con extensión .xls, de Excel 97-03 y con hojas de cálculo más recientes con extensión .xlsx. No obstante, no es compatible con

hojas de cálculo con extensión .ods, de código libre como LibreOffice. Finalmente, su propósito es contar con datos ordenador, viéndose que cada reglón represente una observación y cada columna una variable.

Para su instalación se introducirá en la consola en comando.

install.packages(c("readxl"))

Posteriormente se cargará el paquete:

library(readxl)

4.2.1.2. TSP versión 1.1-10

Este paquete tiene algunos algoritmos y la infraestructura básica para resolver el problema del Travelling Salesman Problem, de esta forma se logra la optimización de las rutas recorridas. Con el uso de varios métodos encuentra buenas soluciones.

Para cargar el paquete se utilizará:

Library (TSP)

4.2.1.3. Shiny versión 1.6.0

El paquete Shiny hace más interactiva la creación de una aplicación con R, siendo la misma de fácil respuesta, con resultados precisos. Haciendo que el usuario trabaje con los datos sin tener que manipular el código. En el caso de nuestra aplicación, el usuario seleccionara las comarcas de provincia por las que desee que entregar producto, el número de unidades e introducir el valor de gasoil en ese momento.

4.2.1.4. Rsconnect versión 0.8.18

Este paquete provee una interfase para el desarrollo de la programación para RPubs, shinyapps.io y RStudio Connect. Soporta contenido como las páginas web, aplicaciones Shiny, y documentos R Markdown, haciendo la publicación del contenido de las aplicaciones externamente.

4.2.2. Microsoft Excel

En este trabajo para la introducción de los valores de precios entre las capitales de comarca, se ha utilizado el software Excel el cual está incluido dentro del paquete Microsoft Office. Este documento será cargado en R lo que permitirá que los valores dentro del documento puedan ser modificados o incluso se pueda cargar otro archivo diferente para hacer uso de la aplicación.

4.2.3. Aplicación web: Shinyapps.io

Shinyapps.io es una herramienta web que permite desarrollar y publicar las aplicaciones creadas. De esta forma el repositorio de la aplicación no se encontrará localmente en el equipo, sino que podrá ser utilizada por cualquier usuario sin la necesidad de instalar los R, siendo así dinámico su uso. Debe tenerse en cuenta que su plan gratuito consta de 25 horas mensuales y 5 aplicaciones activas, para incrementar el número de aplicaciones publicadas, así como las horas activas, será necesario actualizar el plan a uno de pago.

El único requisito necesario es tener una cuenta de Google para acceder. A continuación, ilustraremos los pasos para la publicación de la aplicación desde R.

Ilustración 1:Buscar la web Shinyapps.io para la creación de una cuenta

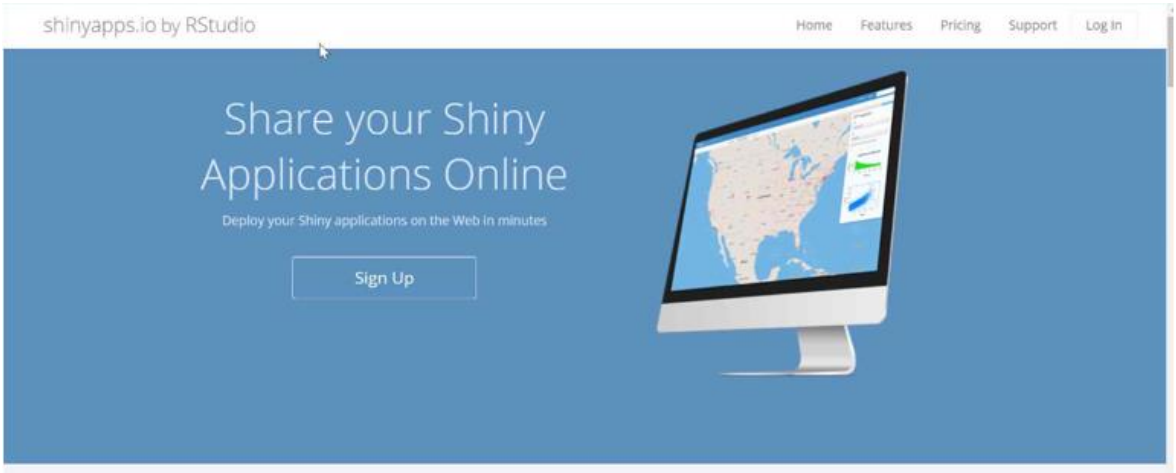


Ilustración 2: Iniciar sesión con Google

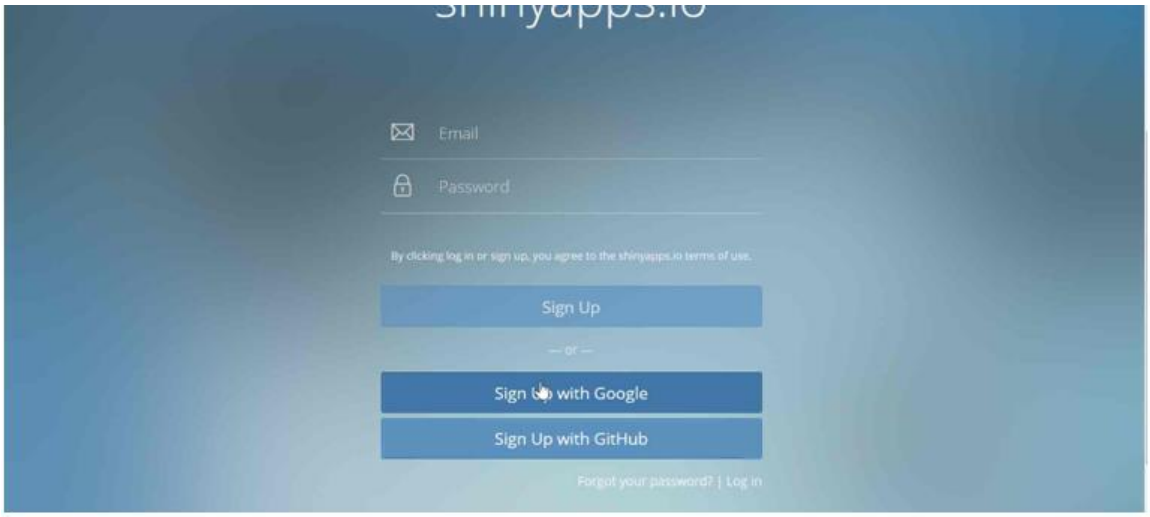
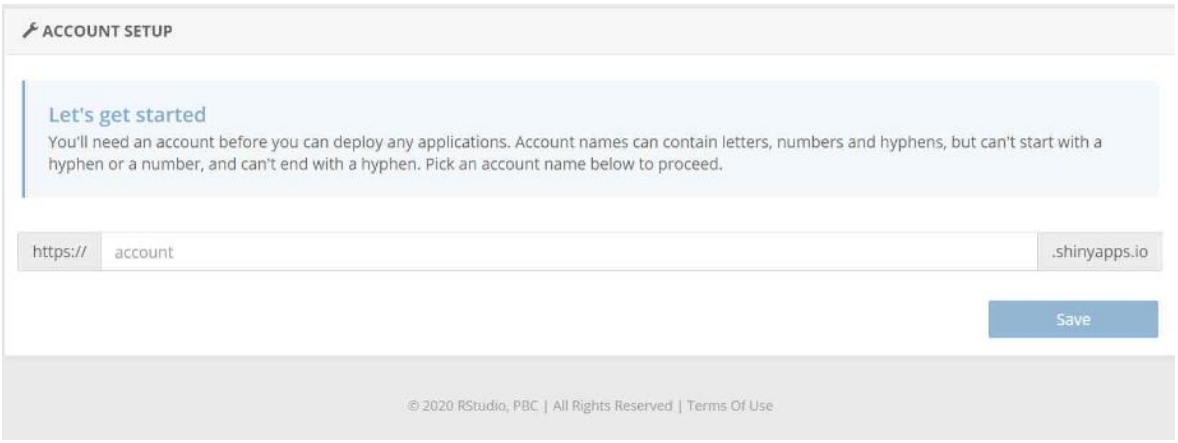


Ilustración 3:Setup de la cuenta



Una vez creada la cuenta entramos dentro de la herramienta RStudio, en el cual instalamos la librería "rsconnect", que permite conectarla app con el repositorio.

Ilustración 4:Publicación

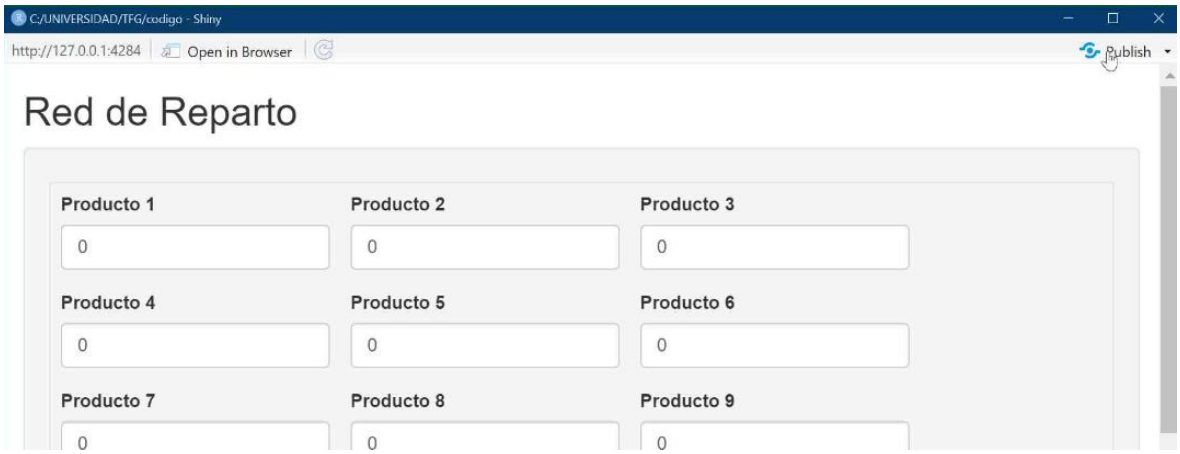


Ilustración 5: Conexión de la cuenta

Ilustración 6: Seguimos los pasos

Los pasos para seguir a continuación permitirán la autorización y el vínculo de la cuanta para subir la aplicación. Será necesario cargar el código del paso 2 mostrado en la consola, una vez mostrado el "secreto".

Ilustración 7:Pasos a seguir para vincular cuenta

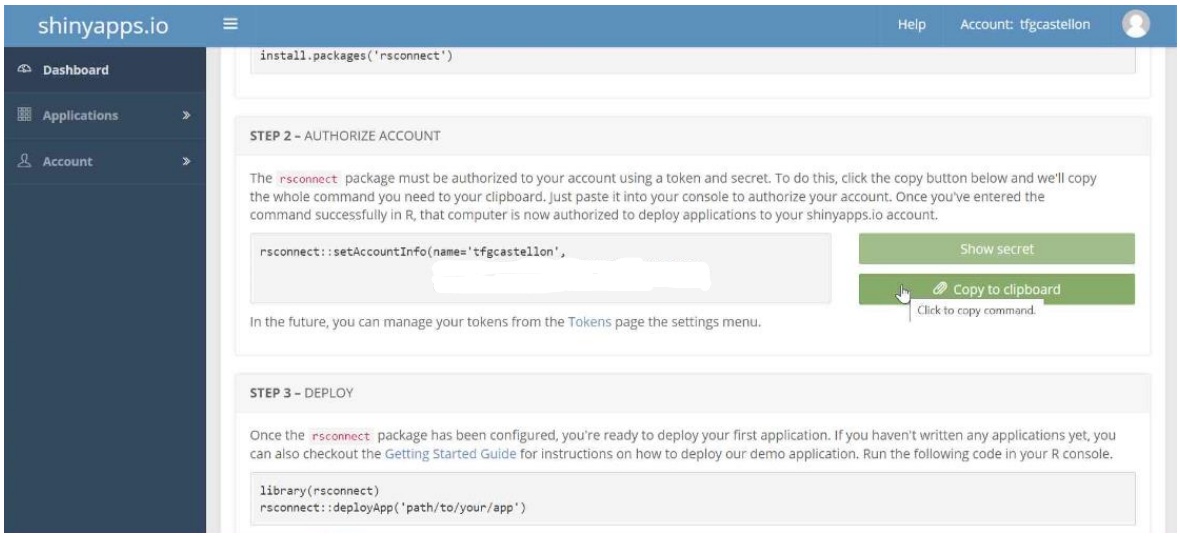
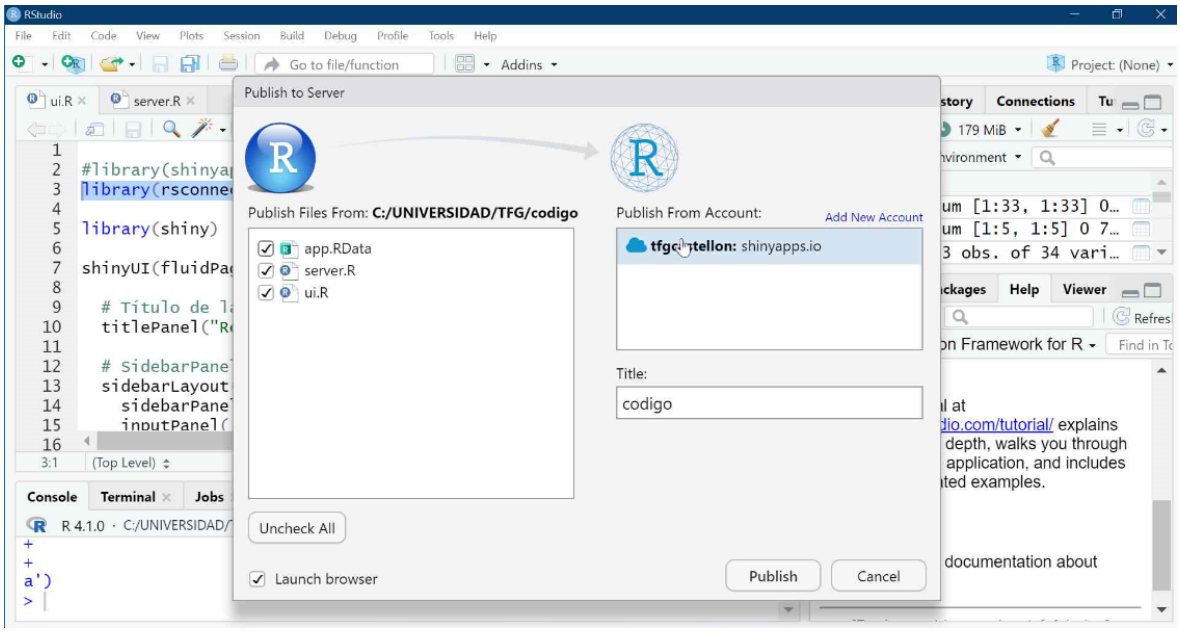


Ilustración 8: Selección de la cuenta en la que se publicara



Al publicarlo cualquier cambio o actualización que se realice, automáticamente se cambiara en el servidor.

https://tfgcastellon.shinyapps.io/Red_de_Reparto/?_ga=2.217340946.523117776.1631266287-1959470887.1630945272

4.3. RED DE DISTRIBUCIÓN. MATRIZ DISTANCIAS DE LA RUTA DE REPARTO.

En relación con lo mencionado anteriormente, este trabajo de fin de grado se basa en la optimización conjunta de una red de reparto, cuyos nodos sean las capitales de comarca de Aragón, en la que se encontrar la ruta de transporte más el coste de compra de los productos.

Seguidamente se muestra una tabla con las capitales de la ruta.

Número	Capital
1	Jaca
2	Sabiñánigo
3	Aínsa
4	Graus
5	Ejea de los Caballeros
6	Huesca
7	Barbastro
8	Monzón
9	Binéfar
10	Tarazona
11	Borja
12	Alagón
13	Illueca
14	La Almunia de Doña Godina
15	Zaragoza
16	Sariñena
17	Fraga
18	Calatayud
19	Cariñena
20	Daroca
21	Belchite
22	Quinto
23	Híjar
24	Caspe
25	Calamocha
26	Utrillas
27	Andorra

28	Alcañiz
29	Valderrobres
30	Albarracín
31	Teruel
32	Cantavieja
33	Mora de Rubielos

Tabla 2: Capitales de la Provincia de Aragón

Las distancias, en Km, se medirán por carretera entre las diferentes capitales de comarca, teniéndose en cuenta las posibles combinaciones, debido a que puede presentarse el caso en la que no se encuentre enlace directo entre sí, sino que se requiera del paso intermedio por otro nodo.

En dicha matriz se encuentran los nombres de los municipios pertenecientes a la red de distribución, así como los valores correspondientes a las distancias mínimas que unen los nodos. Se deberá tener en cuenta que se situaran valores 0 en la diagonal, ya que la distancia entre un nodo y ese mismo nodo se considera nula. Los valores restantes corresponderán, como se mencionó anteriormente, la distancia mínima entre ambos nodos. Dicha matriz es simétrica, el desplazamiento de vehículos en ambos sentidos se permite entre todos los nodos, no siendo el mismo peso de un nodo a otro en ambos sentidos.

En el anexo 1 se mostrará la matriz de distancias entre las cabeceras de comarca.

4.4. MATRIZ DE COSTES DE PRODUCTO

La matriz de costes de productos será introducida en RStudio como un fichero Excel. La tabla contendrá los costes de los 10 productos en cada una de las ciudades dentro de la red de distribución. Los costes son aleatorios, a través de RStudio con el código "rpois", generados aleatoriamente acorde a parámetros indicados. Se decide optar por valores aleatorios ya que dicha matriz podrá ser sustituida por los usuarios, es decir se podrá cargar un fichero Excel diferente permitiendo de esta forma reutilizar el código para cualquier producto que se desee transportar.

La tabla descrita anterior se encuentra en el anexo 2.

4.5. OBTENCIÓN DE LA SOLUCIÓN(SERVER.R) .

Seguidamente se describirán los pasos que se llevarán a cabo en el programa de este trabajo de fin de grado.

Tendremos dos scripts el primero será el server.R , el cual se mostrará en este apartado y el otro, ui.R .

EL archivo server.R nos permite incluir los comandos requeridos, los cuales contienen las instrucciones del funcionamiento de la aplicación. Definiéndose todas las funciones y objetos presentes en él, para posteriormente presentar la solución.

Debido a que se trabajara para crear una aplicación, en la cual se puedan obtener resultados que ofrece el paquete TPS, descrito anteriormente, se incluirán como primeras líneas la carga de los paquetes shiny y TPS, respectivamente.

También en nuestro caso se cargará el paquete readxl ,el cual permitirá la lectura de los archivos de Excel.

```
library(shiny)

library (TSP)

library(readxl)
```

Después se deberá realizar la entrada de datos externos para su ejecución, en este caso, será una matriz en la que se incluirán las distancias entre las cabeceras de comarca de la Comunidad Autónoma de Aragón.

```
Caminos<-
matrix(c(0,18.3,74.5,125.5,77.8,76.8,128.3,145.1,159.1,146.3,134.6,127.9,181.5,17
7.5,150.1,124.8,180.4,210.8,196.9,236.9,199.9,193.7,223.8,198.8,264.8,263.6,256.
4,226.3,261.9,345.2,334.6,334.3,380.1

,18.3,0,56.2,107.2,60,59,110.3,127.1,141.1,128.5,116.8,110.1,163.7,159.7,13
2.3,107,162.6,193,179.1,219.1,182.1,175.9,206,181,247,245.8,238.6,208.5,244.1,3
27.4,316.8,316.5,362.3

,74.5,56.2,0,51,106.6,105.6,54.1,70.9,84.9,175.1,163.4,156.7,210.3,206.3,178
.9,100,130.8,239.6,225.7,265.7,209.5,181.4,211.5,174,293.6,273.2,244.1,201.5,237
.1,374,344.2,309.5,389.7

,125.5,107.2,51,0,81.6,80.6,29.1,45.9,59.9,150.1,138.4,131.7,185.3,181.3,153
.9,75,105.8,214.6,200.7,240.7,184.5,156.4,186.5,149,268.6,248.2,219.1,176.5,212.
1,349,319.2,284.5,364.7
```

,77.8,60,106.6,81.6,0,76.1,52.5,69.3,83.3,68.5,56.8,50.1,103.7,99.7,73.4,49,104.6,133,120.2,160.2,123.2,117,147.1,123,188.1,186.9,179.7,150.5,186.1,268.5,257.9,258.5,303.4

,76.8,59,105.6,80.6,76.1,0,51.5,68.3,82.3,69.5,57.8,51.1,104.7,100.7,73.3,48,103.6,134,120.1,160.1,123.1,116.9,147,122,188,186.8,179.6,149.5,185.1,268.4,257.8,257.5,303.3

,128.3,110.3,54.1,29.1,52.5,51.5,0,16.8,30.8,121,109.3,102.6,156.2,152.2,124.8,45.9,76.7,185.5,171.6,211.6,155.4,127.3,157.4,119.9,239.5,219.1,190,147.4,183,319.9,290.1,255.4,335.6

,145.1,127.1,70.9,45.9,69.3,68.3,16.8,0,14,137.8,126.1,119.4,173,169,120.4,40,59.9,202.3,167.2,207.2,149.5,121.4,151.5,114,235.1,213.2,184.1,141.5,177.1,315.5,284.2,249.5,329.7

,159.1,141.1,84.9,59.9,83.3,82.3,30.8,14,0,151.8,140.1,133.4,187,183,132.4,52,45.9,216.3,179.2,219.2,161.5,133.4,153.8,107.9,247.1,225.2,185.4,135.4,171,327.5,296.2,243.4,326.3

,146.3,128.5,175.1,150.1,68.5,69.5,121,137.8,151.8,0,25,67.2,68.6,73.6,95,117.5,173.1,106.7,94.2,134.2,136.8,138.6,168.7,191.5,162.1,200.5,196.4,199.9,235.5,242.5,271.5,297.4,317

,134.6,116.8,163.4,138.4,56.8,57.8,109.3,126.1,140.1,25,0,42.2,46.9,48.6,70,105.8,161.4,81.9,69.2,109.2,111.8,113.6,143.7,168.7,137.1,175.5,171.4,174.9,210.5,217.5,246.5,272.4,292

,127.9,110.1,156.7,131.7,50.1,51.1,102.6,119.4,133.4,67.2,42.2,0,84.7,49.6,27.8,99.1,151.8,82.9,70.2,110.2,77.6,71.4,101.5,126.5,138.1,141.3,134.1,132.7,168.3,218.5,212.3,235.1,257.8

,181.5,163.7,210.3,185.3,103.7,104.7,156.2,173,187,68.6,46.9,84.7,0,37.7,91,152.7,208.3,38.1,58.3,75.9,100.9,129,139.5,184.1,103.8,161.3,160.5,170.7,206.3,184.2,221.9,261.5,267.4

,177.5,159.7,206.3,181.3,99.7,100.7,152.2,169,183,73.6,48.6,49.6,37.7,0,53.3,133.7,177.3,33.3,20.6,60.6,63.2,91.3,101.8,146.4,88.5,126.9,122.8,133,168.6,168.9,197.9,223.8,243.4

,150.1,132.3,178.9,153.9,73.4,73.3,124.8,120.4,132.4,95,70,27.8,91,53.3,0,80.4,124,86.6,46.8,86.8,49.8,43.6,73.7,98.7,114.7,113.5,106.3,104.9,140.5,195.1,184.5,207.3,230

,124.8,107,100,75,49,48,45.9,40,52,117.5,105.8,99.1,152.7,133.7,80.4,0,55.6,167,127.2,167.2,109.5,81.4,111.5,74,195.1,173.2,144.1,101.5,137.1,275.5,244.2,209.5,289.7

,180.4,162.6,130.8,105.8,104.6,103.6,76.7,59.9,45.9,173.1,161.4,151.8,208.3,177.3,124,55.6,0,210.6,167.2,207.2,124.6,96.5,107.9,62,220.9,188.3,139.5,89.5,125.1,297,259.3,197.5,280.4

,210.8,193,239.6,214.6,133,134,185.5,202.3,216.3,106.7,81.9,82.9,38.1,33.3,86.6,167,210.6,0,53.5,37.8,96.1,124.2,134.7,179.3,65.7,123.2,133.7,165.9,201.5,146.1,183.8,227.2,229.3

,196.9,179.1,225.7,200.7,120.2,120.1,171.6,167.2,179.2,94.2,69.2,70.2,58.3,20.6,46.8,127.2,167.2,53.5,0,40,42.6,70.7,81.2,125.8,67.9,106.3,102.2,112.4,148,148.3,177.3,203.2,222.8

,236.9,219.1,265.7,240.7,160.2,160.1,211.6,207.2,219.2,134.2,109.2,110.2,75.9,60.6,86.8,167.2,207.2,37.8,40,0,82.6,110.7,121.2,165.8,27.9,85.4,95.9,145.9,181.5,108.3,146,189.4,191.5

,199.9,182.1,209.5,184.5,123.2,123.1,155.4,149.5,161.5,136.8,111.8,77.6,100.9,63.2,49.8,109.5,124.6,96.1,42.6,82.6,0,28.1,38.6,83.2,96.3,63.7,59.6,69.8,105.4,172.4,134.7,160.6,180.2

,193.7,175.9,181.4,156.4,117,116.9,127.3,121.4,133.4,138.6,113.6,71.4,129,91.3,43.6,81.4,96.5,124.2,70.7,110.7,28.1,0,30.1,55.1,124.4,91.8,62.7,61.3,96.9,200.5,162.8,163.7,208.3

,223.8,206,211.5,186.5,147.1,147,157.4,151.5,153.8,168.7,143.7,101.5,139.5,101.8,73.7,111.5,107.9,134.7,81.2,121.2,38.6,30.1,0,45.9,134.9,84.7,32.6,31.2,66.8,193.4,155.7,133.6,201.2

,198.8,181,174,149,123,122,119.9,114,107.9,191.5,168.7,126.5,184.1,146.4,98.7,74,62,179.3,125.8,165.8,83.2,55.1,45.9,0,179.5,129.6,77.5,27.5,63.1,238.3,200.6,135.5,218.4

,264.8,247,293.6,268.6,188.1,188,239.5,235.1,247.1,162.1,137.1,138.1,103.8,88.5,114.7,195.1,220.9,65.7,67.9,27.9,96.3,124.4,134.9,179.5,0,57.5,109.6,159.6,195.2,80.4,118.1,161.5,163.6

,263.6,245.8,273.2,248.2,186.9,186.8,219.1,213.2,225.2,200.5,175.5,141.3,161.3,126.9,113.5,173.2,188.3,123.2,106.3,85.4,63.7,91.8,84.7,129.6,57.5,0,52.1,102.1,137.7,108.7,71,104,116.5

,256.4,238.6,244.1,219.1,179.7,179.6,190,184.1,185.4,196.4,171.4,134.1,160.5,122.8,106.3,144.1,139.5,133.7,102.2,95.9,59.6,62.7,32.6,77.5,109.6,52.1,0,50,85.6,160.8,123.1,101,168.6

,226.3,208.5,201.5,176.5,150.5,149.5,147.4,141.5,135.4,199.9,174.9,132.7,170.7,133,104.9,101.5,89.5,165.9,112.4,145.9,69.8,61.3,31.2,27.5,159.6,102.1,50,0,35.6,210.8,173.1,108,190.9

,261.9,244.1,237.1,212.1,186.1,185.1,183,177.1,171,235.5,210.5,168.3,206.3,168.6,140.5,137.1,125.1,201.5,148,181.5,105.4,96.9,66.8,63.1,195.2,137.7,85.6,35.6,0,227.1,189.4,94.1,177

,345.2,327.4,374,349,268.5,268.4,319.9,315.5,327.5,242.5,217.5,218.5,184.2,168.9,195.1,275.5,297,146.1,148.3,108.3,172.4,200.5,193.4,238.3,80.4,108.7,160.8,210.8,227.1,0,37.7,133,83.2

,334.6,316.8,344.2,319.2,257.9,257.8,290.1,284.2,296.2,271.5,246.5,212.3,221.9,197.9,184.5,244.2,259.3,183.8,177.3,146,134.7,162.8,155.7,200.6,118.1,71,123.1,173.1,189.4,37.7,0,95.3,45.5

,334.3,316.5,309.5,284.5,258.5,257.5,255.4,249.5,243.4,297.4,272.4,235.1,261.5,223.8,207.3,209.5,197.5,227.2,203.2,189.4,160.6,163.7,133.6,135.5,161.5,104,101,108,94.1,133,95.3,0,82.9

,380.1,362.3,389.7,364.7,303.4,303.3,335.6,329.7,326.3,317,292,257.8,267.4,243.4,230,289.7,280.4,229.3,222.8,191.5,180.2,208.3,201.2,218.4,163.6,116.5,168.6,190.9,177,83.2,45.5,82.9,0),byrow=TRUE,ncol=33)

dimnames(Caminos)<-
list(c("JACA","SABIÑANIGO","AÍNSA","GRAUS","EJEA.DE.LOS.CABALLEROS","HUESCA","BARBASTRO","MONZÓN","BINÉFAR","TARAZONA","BORJA","ALAGÓN","ILLUECA","LA.ALMUNIA.DE.DOÑA.GODINA","ZARAGOZA","SARIÑENA","FRAGA","CALATAYUD","CARIÑENA","DAROCA","BELCHITE","QUINTO","HÍJAR","CASPE","CALAMOCHA","UTRILLAS","ANDORRA","ALCAÑIZ","VALDERROBRES","ALBARRACÍN","TERUEL","CANTAVIEJA","MORA.DE.RUBIELOS")

,c("JACA","SABIÑANIGO","AÍNSA","GRAUS","EJEA.DE.LOS.CABALLEROS","HUESCA","BARBASTRO","MONZÓN","BINÉFAR","TARAZONA","BORJA","ALAGÓN","ILLUECA","LA.ALMUNIA.DE.DOÑA.GODINA","ZARAGOZA","SARIÑENA","FRAGA","CALATAYUD","CARIÑENA","DAROCA","BELCHITE","QUINTO","HÍJAR","CASPE","CALAMOCHA","UTRILLAS","ANDORRA","ALCAÑIZ","VALDERROBRES","ALBARRACÍN","TERUEL","CANTAVIEJA","MORA.DE.RUBIELOS"))

A continuación, se asignará el fichero Excel con el que se podrá subir datos de costes de productos:

```
MatrizPrecios1<-
read_excel("C:/UNIVERSIDAD/TFG/Costeproductos.xlsx",col_names=TRUE)

MatrizPrecios1<as.matrix(MatrizPrecios1)

is.matrix(MatrizPrecios1)
```

Con el comando "as.matrix()" utilizado, se convertirán los datos de Excel en una matriz, obteniéndose objetos con el atributo de matriz para ser en otro momento.

```
MatrizPrecios1<<-sapply(seq(1,33),f)

write.xlsx2(MatrizPrecios1,"C:/UNIVERSIDAD/TFG/MatrizPrecios1.xlsx",sheet.na
me="Precios")

dimnames(MatrizPrecios1)[[2]]<<-dimnames(Caminos)[[2]]
```

La función "write.xlsx2()" se utilizará para exportar los valores de RStudio a un libro de Excel. En este comando se deberá especificar la ruta al archivo de salida, así como el nombre de la hoja._Siendo la función shinyServer(), la que utilice los inputs para generar otros objetos que posteriormente se utilizaran para ofrecer soluciones, devolviéndolas, creando outputs.

```
shinyServer(function(input, output) {
```

La función reactive(), recalcula los resultados de forma casi inmediata, si se modifican las variables de entrada/inputs.

```
slidervalues<-reactive({
```

Se designará la función con la que se trabajará, se insertará un dummy city, para que realice un corte o "cut" en el tour hamiltoniano:

```
Fprecios<-function(x){

Matriz<-Caminos[c(input$variable),c(input$variable)]

CiudadesRepartoTSP<-TSP(Matriz, labels = NULL)

CRtsp <- insert_dummy(CiudadesRepartoTSP, label = "cut")
```

Entrada relativa al algoritmo empleado :

```
 method1<-"nearest_insertion"
```

```
method2<-"farthest_insertion"
```

Se definirá el origen de la ruta, teniendo en cuenta dos posibles escenarios. El "results1", estará ligado al método del vecino más cercano. De la misma manera el "results 2" analizara con el método el vecino más lejano. En el apartado (4.1.3) de "Algoritmos para su resolución", se explicarán estos métodos.

El "initialtour" se definirá el origen de la ruta:

```
initialtour<-as.integer(which(labels(Matriz)[[1]]==dimnames(Matriz)[[2]][x]))


results1<-solve_TSP((CRtsp),method1,control=list(start=initialtour))
  results2<-solve_TSP((CRtsp),method2,control=list(start=initialtour))
  if(tour_length(results1)<=tour_length(results2))
  V<-as.integer(results1)
  else
  V<-as.integer(results2)
```

De esta forma, una vez el algoritmo calcule los dos métodos compara los dos resultados, arrojando la aplicación el menor de ellos, que estará asociado al menor coste de ruta.

Designación de la matriz de precios, previamente cargada, ya que nuestro objetivo es encontrar la mejor ruta de reparto, en la cual se elegirá el mejor almacén de salida que determine el menor precio :

```
  Precios<-matrix(NA,nrow=10,ncol=33)
 Precios<-MatrizPrecios1[,c(input$variable)]
```

Input relativo de los productos que se distribuirán:

```
  Unidades<-
c(input$Producto1,input$Producto2,input$Producto3,input$Producto4,input$Producto5
,input$Producto6,input$Producto7,input$Producto8,input$Producto9,input$Producto10
)
```

Se calcula la ciudad de origen de reparto para realizar el tour hamiltoniano.

```
OrigenReparto<-
input$variable[which.min(sapply(seq(1,length(input$variable)),Fprecios))]
if (length(input$variable)<2) stop("")
```

```
 else {
   Matriz<-Caminos[c(input$variable),c(input$variable)]
 CiudadesRepartoTSP<-TSP(Matriz, labels = NULL)
```

Cálculo que realizara la aplicación, en el que se contempla el número de unidades por el precio del producto, anteriormente mencionado en la matriz, sumado al valor de la distancia multiplicada por el precio de la gasolina en ese momento, dicho precio introducido manualmente por el usuario.

```
    sum(Unidades*Precios[,x])+  tour_length(CRtsp, V)*input$Cgasolina}
rbind(input$variable,sapply(seq(1,length(input$variable)),Fprecios))
```

Para arrojar los datos en formato de tabla se ejecutará la función renderTable().

En este código se muestra los outputs o textos que figuran en la aplicación, los cuales acompañaran las salidas de valores para definir cada salida y que el usuario sepa que valor está viendo:

```
 output$values<- renderTable({
  Tabla<-data.frame(slidervalues())
  names(Tabla)<-"Ruta de reparto"
  Tabla
 })
output$Distancia<- renderText({
 "Longitud de la ruta(Km)"
})
output$Capitales<- renderText({
 "Coste de la Ruta"
})
output$valuess<- renderPrint({
 slidervalues1()
})
 slidervaluesPrecios()
```

```
output$valuess2<-renderTable({
 D<-data.frame( slidervaluesPrecios())
 names(D)<-NULL
 D}) })
```

4.6. USER INTERFASE (UI.R)

Ui.R ,es el lugar donde se define que es lo que veremos en la aplicación , en el cual se definirá la salida HTML que se desea, incluyendo la carga de paquetes shiny y shinyapps. Presentará una serie de comandos que controlan el diseño y el aspecto que tendrá la solución. Pudiendo gestionar los inputs y outputs de títulos y textos, que se visualizan.

Las primeras líneas del código se destinarán a la carga de los paquetes shiny y rsconnect que permiten la generación de la app y su visualización online.

```
library(rsconnect)
library(shiny)
```

Se iniciará con la introducción del título de la aplicación con el comando "titlePanel()", en el cual se muestra el nombre de la misma.

```
shinyUI(fluidPage(
  titlePanel("Red de Reparto"),
```

Con la línea siguiente de código se podrán cargar los precios de los productos a distribuir recogidos en una Excel, facilitando su uso a la hora de una modificación en los datos o en el caso de que se desee utilizar un fichero diferente.

```
fileInput("MatrizPrecios1", "Cargar fichero de precios", accept = ".xlsx"),
```

Se muestra a continuación como se incluyen un "numericInput" para los 10 productos , que permite la introducción del número de unidades que se necesiten transportar de cada uno de ellos ,tomándose como máximo valor 1000 unidades.

```
sidebarLayout(
   sidebarPanel(
   inputPanel(
   numericInput("Producto1","Producto 1",value=0,min=0,max=1000,step=1),
```

```
numericInput("Producto2","Producto 2",value=0,min=0,max=1000,step=1),
numericInput("Producto3","Producto 3",value=0,min=0,max=1000,step=1),
numericInput("Producto4","Producto 4",value=0,min=0,max=1000,step=1),
numericInput("Producto5","Producto 5",value=0,min=0,max=1000,step=1),
numericInput("Producto6","Producto 6",value=0,min=0,max=1000,step=1),
numericInput("Producto7","Producto 7",value=0,min=0,max=1000,step=1),
numericInput("Producto8","Producto 8",value=0,min=0,max=1000,step=1),
numericInput("Producto9","Producto 9",value=0,min=0,max=1000,step=1),
numericInput("Producto10","Producto10",value=0,min=0,max=1000,step=1),
```

Se añadirá una barra para introducir el coste de combustible por km en ese momento para que se pueda computar correctamente el coste total.

```
sliderInput("Cgasolina","Coste gasolina por km",value=0.23,min=0,max=1,step=0.01)),
```

En el siguiente comando se introducirán los datos de los almacenes en las distintas ciudades visitadas en la ruta de reparto. Estas podrán ser seleccionadas por el usuario en un panel ordenadas alfabéticamente, pudiendo escogerse la ruta de paso.

```
checkboxGroupInput("variable", "Seleccionar las ciudades que se desea incluir en la ruta",
choices=c("AÍNSA"="AÍNSA","ALAGÓN"="ALAGÓN","ALBARRACÍN"="ALBARRACÍN",
                                "ALCAÑIZ"="ALCAÑIZ",
                                "LA ALMUNIA DE DOÑA GODINA"="LA.ALMUNIA.DE.DOÑA.GODINA",
                                "ANDORRA"="ANDORRA","BARBASTRO"="BARBASTRO",
"BELCHITE"="BELCHITE","BINÉFAR"="BINÉFAR","BORJA"="BORJA",
                            "CALAMOCHA"="CALAMOCHA","CALATAYUD"="CALATAYUD",
                            "CANTAVIEJA"="CANTAVIEJA","CARIÑENA"="CARIÑENA",
                                "CASPE"="CASPE","DAROCA"="DAROCA",
                                "EJEA DE LOS CABALLEROS"="EJEA.DE.LOS.CABALLEROS",
```

```
"FRAGA"="FRAGA","GRAUS"="GRAUS","HÍJAR"="HÍJAR",
"HUESCA"="HUESCA","ILLUECA"="ILLUECA","JACA"="JACA",
"MONZÓN"="MONZÓN","MORA DE
RUBIELOS"="MORA.DE.RUBIELOS",
"QUINTO"="QUINTO","SABIÑANIGO"="SABIÑANIGO",
"SARIÑENA"="SARIÑENA","TARAZONA"="TARAZONA",
"TERUEL"="TERUEL","UTRILLAS"="UTRILLAS",
"VALDERROBRES"="VALDERROBRES","ZARAGOZA"="ZARAGOZA"),
selected=c("ZARAGOZA"="ZARAGOZA",
"TERUEL"="TERUEL","HUESCA"="HUESCA"),
inline=TRUE) ,
selectInput("Method",label="Algoritmo",
choices=c("nearest_insertion", "farthest_insertion")
```

Se introducen los métodos que se utilizaran para el cálculo del problema. Estos métodos se explican en el apartado anterior Obtención de la solución(4.5).Seguidamente con el width se decidirá la anchura que ocupara la aplicación en este caso se ha decidido como valor 12.

```
Width=12
```

Una vez creado la interfaz se programarán las salidas u outputs que se desea que la aplicación devuelva. Con el siguiente comando se tendrá un "textOutput" que hace posible que se incluya texto en la interfaz, y un "verbatimTextOutput" , que lo capta.

Con el "tableOutput(values)" será posible incluir una tabla que se rellenará en sentido descendente, con el orden de la ruta de reparto a seguir. El dato inicial de la tabla será la primera ciudad por la que se comienza el reparto. Se incluirá con el comando "strong", el concepto "Costes de compra en almacén y transporte" relacionado a los valores finales en la aplicación.

También en la app se mostrará con el comando "tableOutput(valuess2)", otra tabla que dará los valores de costes referentes a cada una de las ciudades incluidas en el reparto. Se leerá de izquierda a derecha tomando como referencia la tabla anterior, es decir, la primera ciudad de la ruta corresponderá al primer valor de la tabla de

costes, la segunda ciudad ordenada de forma descendente será la siguiente y así sucesivamente.

```
mainPanel(textOutput("Distancia"),
verbatimTextOutput("valuess"),
tableOutput("values"),
textOutput("Capitales"),
strong("Costes de compra en almacén y transporte"),
tableOutput("valuess2")
```

4.7. INTERFAZ DE LA APP

En este capítulo se pretende mostrar el aspecto y la distribución final de la aplicación que resuelve el problema propuesto. Asimismo, se distinguirá entre los inputs y outputs de esta.

Inicialmente se presentarán los inputs de la aplicación. En la imagen siguiente se verá como el usuario puede ingresar el valor deseado referente a la cantidad de unidades, de los 10 productos, que se distribuirán entre los municipios. Se incluye un "slider input" en el cual se deberá incluir el coste de gasolina y un "Browser", en el que el usuario buscará el archivo con la matriz de precios que desee cargar.

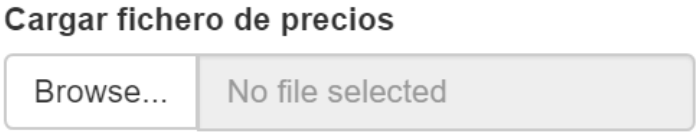


Ilustración 9: Carga de ficheros en la app

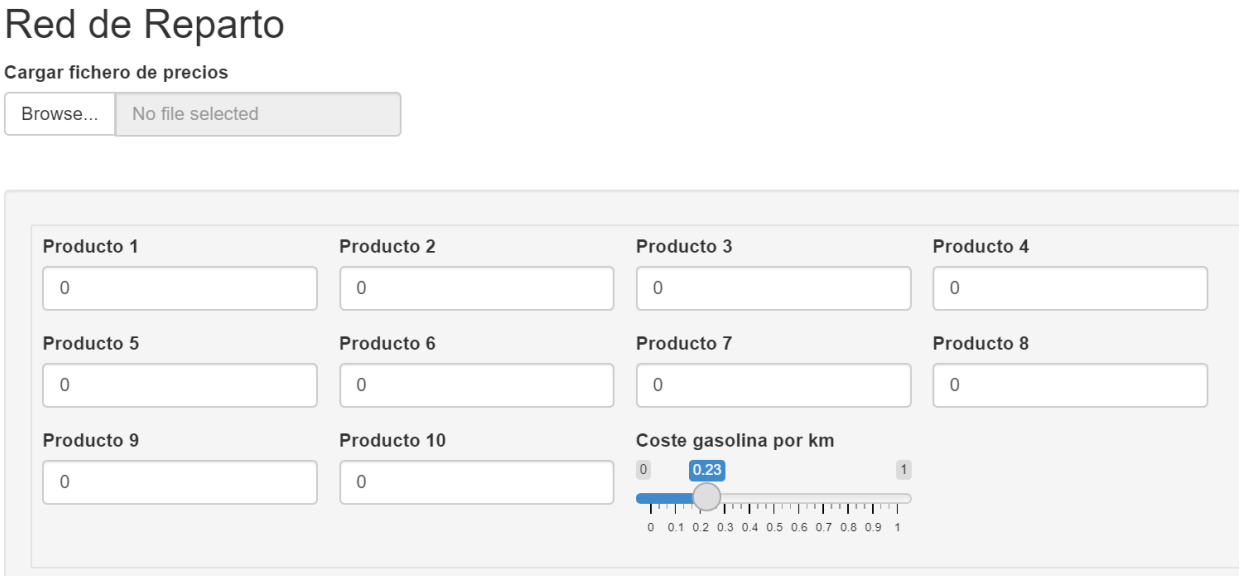


Ilustración 10: Interfaz de los inputs de la app

Posteriormente se seleccionarán los municipios que estarán incluidos en la ruta.

EL consumo por kilómetro de gasolina se mostrará de la siguiente forma en la aplicación. Se tomará por defecto el valor de 0.23 por KM ya que el camión de reparto escogido como ejemplo , consume 22,28€/100 km. Dicha valor podrá ser cambiado por el usuario.

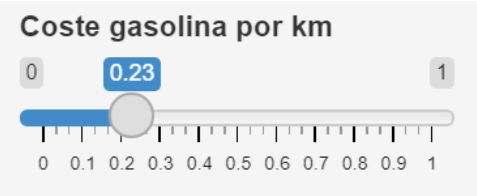


Ilustración 11: Coste de gasolina por Km

Se indicará en un panel, los municipios que están dentro de la red de distribución y en la matriz de distancias.

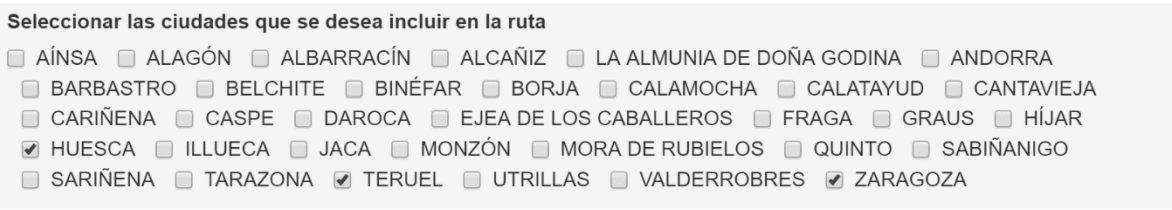
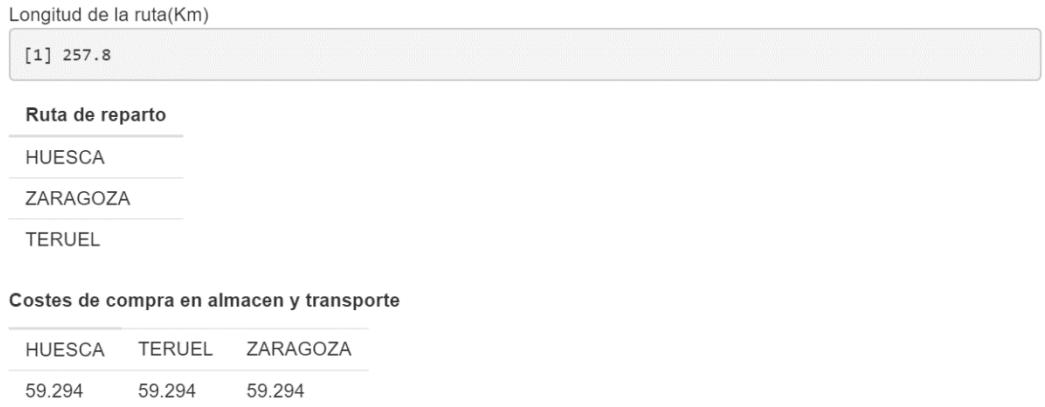


Ilustración 12:Panel de selección de municipios

A continuación, se mostrarán los outputs de la aplicación, una vez visto todas las entradas.


Longitud de la ruta(Km)

[1] 257.8

Ruta de reparto

HUESCA
ZARAGOZA
TERUEL

Costes de compra en almacen y transporte

HUESCA	TERUEL	ZARAGOZA
59.294	59.294	59.294

Ilustración 13: Interfaz Outputs de la app

Una salida que se ve es la longitud medida en kilómetros de la ruta de reparto, que se obtiene en los cálculos.

También vemos una tabla con los municipios incluidos en la ruta, con el orden en que serán visitados, podemos ver un ejemplo en la siguiente ilustración:


Ruta de reparto

HUESCA
ZARAGOZA
TERUEL

Ilustración 14: Ejemplo de rutas de reparto

Finalmente, se muestra la tabla de costes finales para cada municipio como si éste fuera origen de la ruta. Para cada ciudad se estima el coste tomando como almacén de origen el municipio de referencia, su cálculo corresponde al coste de compra de los productos en dicha ciudad más el coste de transporte de la ruta tomando como origen el municipio. Estos valores son estimados en la optimización, y

la ruta óptima final corresponde a aquel municipio con menor coste de compra y transporte.

Costes de compra en almacen y transporte

HUESCA	TERUEL	ZARAGOZA
59.294	59.294	59.294

Ilustración 15: Tabla de resultados Costes de ruta

Para detallar el cálculo de los costes por la app se incluirá un ejemplo del cálculo, en el que se incluyen cantidades aleatorias en los 10 productos y los costes de transportes, que se incluyen en el fichero de precios.

Primeramente, se carga el fichero de precios y se introduce manualmente la cantidad de producto a distribuir y el coste de gasolina, en función del consumo.

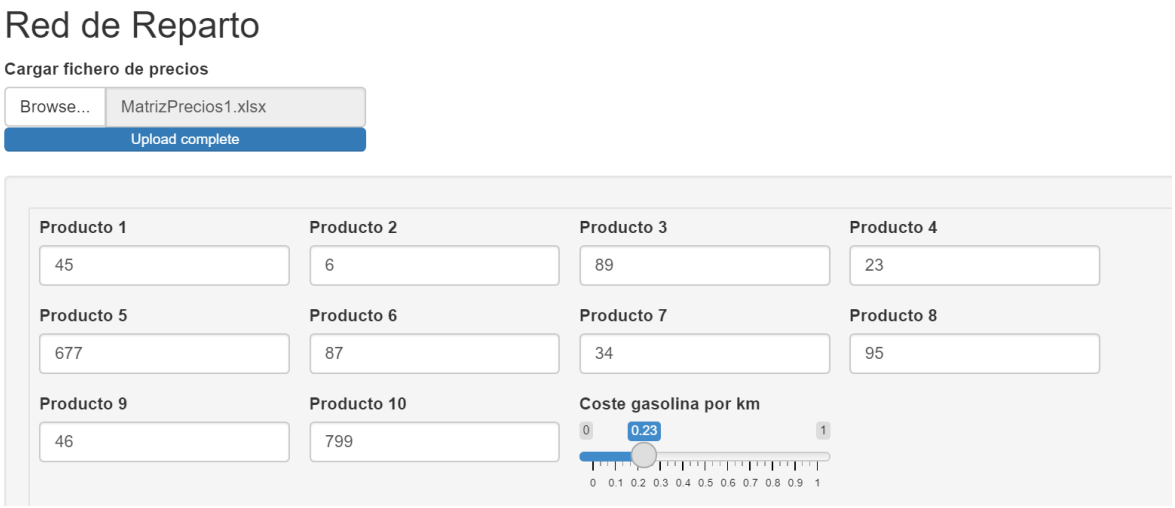


Ilustración 16: Valores ejemplo de producto y gasoil

Se seleccionan las ciudades a las que se desea realizar el reparto.

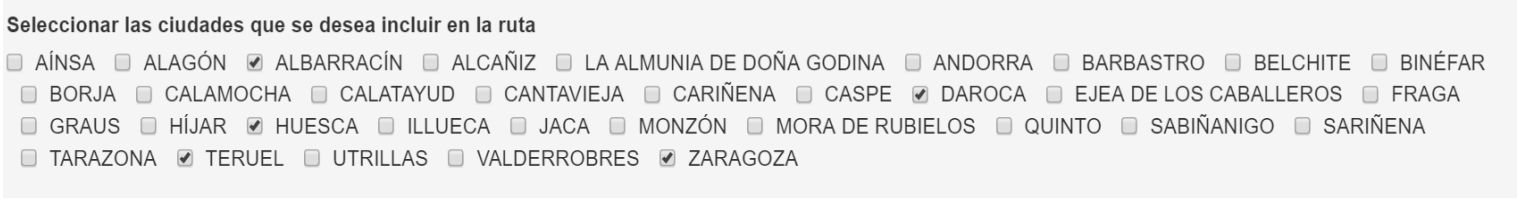


Ilustración 17: Selección aleatoria de ciudades de reparto.

La aplicación arroja los costes totales de cada cabecera de comarca incluida en la ruta, se tomará como almacén de origen el de menor coste ,trazando la mejor ruta de reparto con menor coste.

Costes de compra en almacen y transporte

ALBARRACÍN	DAROCA	HUESCA	TERUEL	ZARAGOZA
89054.403	92254.403	83530.403	96200.403	94626.403

Ilustración 18: Resultado de costes de la ruta de reparto

De esta forma se mostrará la longitud de la ruta a seguir en km y la ruta a seguir en función del almacén de origen escogido.

La app procederá de la misma forma para distintos valores que se incluyan y a las ciudades que se deseen visitar.

Longitud de la ruta(Km)

```
[1] 306.1
```

Ruta de reparto
HUESCA
ZARAGOZA
DAROCA
ALBARRACÍN
TERUEL

Ilustración 19: Ruta de reparto a seguir en función del almacén de origen escogido

5. CONCLUSIONES

Las empresas de transporte de mercancías hoy en día se ven obligadas a optimizar recursos para poder ser competitivas dentro del mercado. Haciendo de esta forma , rutas más eficientes ,siendo esta mejora la que permita la reducción de costes en su cadena de valor.

Trabajando el problema del viajante para la optimización de costes de ruta en las capitales de comarca de la comunidad de Aragón, se han propuesto varios algoritmos heurísticos para su resolución. Debido a la complejidad en el cálculo de soluciones optimas, métodos como el vecino más cercano y el vecino más lejano, se incluyen dentro del cálculo de la solución final.

En general, aunque estos dos métodos están diseñados para resolver el problema, no aseguran una solución óptima, suelen brindar buenas soluciones y tiene un tiempo de cálculo eficiente, siendo el adecuando para cumplir lo propuesto inicialmente.

He de destacar que la creación de la herramienta web permitirá una mayor accesibilidad e interacción de los usuarios con la aplicación, gracias a Shinyapps.io. El usuario podrá elegir una serie de parámetros iniciales como la cantidad de producto que se distribuirá y el coste de gasolina .Además, elegirán las ciudades que desean que se haga el reparto, de esta forma el programa arrojara cual será el mejor almacén de origen para la ruta de reparto, mostrándose finalmente el coste de transporte total asociado a cada una de las ciudades en elegidas.

Con lo abordado en este proyecto concluyo que se han cumplido los objetivos marcados en el inicio:

- Investigación sobre el problema del viajante y la ruta óptima, para así conocer los distintos algoritmos heurísticos para alcanzar una solución óptima.
- Utilización de la aplicación R Studio como herramienta principal para la resolución del problema planteado.
- Cálculo del coste total de transporte para el análisis de la ruta optima.

Pudiendo abarcarse en el futuro la inclusión de las capacidades de los almacenes en el análisis de la ruta de reparto, permitiendo seleccionar el almacén no solo por el coste de transporte sino por la capacidad disponible en ese momento. Haciendo énfasis en la capacidad del mismo ,tendiendo esta última mayor influencia ya que, si la capacidad no es la adecuada para el pedido, pero su coste es menor no podrá ser

seleccionado, y deberá escogerse un almacén de coste mayor con capacidad suficiente.

6. BIBLIOGRAFÍA

[1] G. Tirado Domínguez, Á. Felipe Ortega, M. T. Ortuño Sánchez, y C. e-libro, *El doble problema del viajante con múltiples pilas*. Madrid: Universidad Complutense de Madrid, Servicio de Publicaciones, 2010.

[2] «222808009.pdf». Accedido: jul. 30, 2021. [En línea]. Disponible en: https://core.ac.uk/download/pdf/222808009.pdf

[3] «ACTAS_MATVI_2008.pdf». Accedido: jul. 04, 2021. [En línea]. Disponible en: http://www.lcc.uma.es/~afdez/ACTAS_MATVI_2008.pdf#page=27

[4] A. rvaquerizo, «Aprendiendo shiny. server.R ui.R», *Análisis y Decisión*, feb. 02, 2015. https://analisisydecision.es/aprendiendo-shiny-server-r-ui-r/ (accedido jul. 30, 2021).

[5] Á. A. Ambite y L. de P. Hernández, «El Problema del Viajante de Comercio: análisis teórico y estrategias de resolución», p. 7.

[6] F. P. G. Márquez, «Optimización de rutas en una empresa de distribución de mercancías», *Ind. Manag.*, p. 10, 2007.

[7] A. Schrijver, «On the History of Combinatorial Optimization (Till 1960)», en *Handbooks in Operations Research and Management Science*, vol. 12, Elsevier, 2005, pp. 1-68. doi: 10.1016/S0927-0507(05)12001-5.

[8] «Librería XLSX en R». http://rstudio-pubs-static.s3.amazonaws.com/282636_7933fef0d33b48248d0f175a6c0778bb.html (accedido jul. 30, 2021).

[9] «Infantes Durán Miguel TFG.pdf». Accedido: jul. 04, 2021. [En línea]. Disponible en: https://idus.us.es/bitstream/handle/11441/77531/Infantes%20Dur%c3%a1n%20Miguel%20TFG.pdf?sequence=1&isAllowed=y

[10] S. R. García, «Estructuras de Datos», p. 64.

[11] M. L. Stockdale, «El problema del viajante: un algoritmo heurístico y una aplicación.», p. 114.

[12] J. M. Daza, J. R. Montoya, y F. Narducci, «RESOLUCIÓN DEL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON LIMITACIONES DE CAPACIDAD UTILIZANDO UN PROCEDIMIENTO METAHEURÍSTICO DE DOS FASES», *Rev. EIA*, n.º 12, pp. 23-38, dic. 2009.

[13] «rsconnect.pdf». Accedido: jul. 13, 2021. [En línea]. Disponible en: https://cran.r-project.org/web/packages/rsconnect/rsconnect.pdf

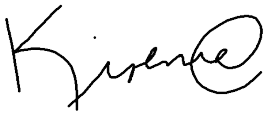
[14] «TAZ-TFG-2020-1932.pdf». Accedido: jul. 04, 2021. [En línea]. Disponible en: https://zaguan.unizar.es/record/98156/files/TAZ-TFG-2020-1932.pdf

[15] M. Tupia y D. Mauricio, «UN ALGORITMO VORAZ PARA RESOLVER EL PROBLEMA DE LA PROGRAMACIÓN DE TAREAS DEPENDIENTES EN MÁQUINAS DIFERENTES», p. 10, 2004.

Relación de documentos

(X) Memoria	38	páginas
(X) Anexos	2	páginas

La Almunia, a 21 de Septiembre de 2021



Firmado: Kirenia Castellón Carballo Carballo